SEARCH RESULTS [PDF Full-Text (872 KB)] PREVIOUS NEXT DOWNLOAD CITATIC

Access control in wide-area networks

Hiltunen, M.A. Schlichting, R.D.
Dept. of Comput. Sci., Arizona Univ., Tucson, AZ ;

This paper appears in: **Distributed Computing Systems, 1997., Proceedi**
of the 17th International Conference on
Meeting Date: 05/27/1997 -05/30/1997
Publication Date: 27-30 May 1997
Location: Baltimore, MD , USA
On page(s): 330-337
References Cited: 30
Number of Pages: xvii+596
INSPEC Accession Number: 5622806

Abstract:
Access control involves maintaining information about which users can access system resources and ensuring that access is restricted to authorized users. I wide-area networks such as the Internet, implementing access control is diffi since resources may be replicated, the task of managing access rights may b distributed among multiple sites, and events such as host failures, host recov and network partitions must be dealt with. This paper explores the problem o access control in such an environment, and in particular the inherent tradeoff between security, availability, and performance. Techniques for dealing with a control in the presence of partitions are presented and used as the basis for a algorithm that allows application control over these tradeoffs

Index Terms:
authorisation computer network reliability performance evaluation wide-area network Internet access control access rights application control authorized users availabili host failures host recoveries multiple sites network partitions performance replicat resources security system resources wide area networks

Documents that cite this document
Select link to view other documents in the database that cite this one.

SEARCH RESULTS [PDF Full-Text (872 KB)] PREVIOUS NEXT DOWNLOAD CITATION

Home | Log-out | Journals | Conference Proceedings | Standards | Search by Author | Basic Search | Advanced
Join IEEE | Web Account | New this week | OPAC Linking Information | Your Feedback | Technical Support | Ema
No Robots Please | Release Notes | IEEE Online Publications | Help | FAQ | Terms | Back to Top

IEEE Xplore Citation

Access Control in Wide-Area Networks*

Matti A. Hiltunen and Richard D. Schlichting
Department of Computer Science
University of Arizona
Tucson, AZ 85721, USA

Abstract

Access control involves maintaining information about which users can access system resources and ensuring that access is restricted to authorized users. In wide-area networks such as the Internet, implementing access control is difficult, since resources may be replicated, the task of managing access rights may be distributed among multiple sites, and events such as host failures, host recoveries, and network partitions must be dealt with. This paper explores the problem of access control in such an environment, and in particular, the inherent tradeoff between security, availability, and performance. Techniques for dealing with access control in the presence of partitions are presented and used as the basis for an algorithm that allows application control over these tradeoffs.

1 Introduction

It is important in any computing system to be able to coordinate and regulate the access of users or programs to various system resources. For example, file systems typically incorporate *access control mechanisms* to enforce policies that specify the rights of each user for each file. In wide-area networks such as the Internet, implementing such mechanisms is challenging when compared with either centralized or local-area solutions, for a number of reasons. One is that the resources themselves may be replicated onto several hosts to improve performance and availability. Another is that the task of managing the access rights for each object may be distributed among more than one managing site. Finally, in these types of environments, events such as host failures, host recoveries, and network partitions can be frequent and must be tolerated by the access control protocol.

The problem of access control in distributed environments has been explored in a number of systems. For example, Amoeba [16] and the Andrew file system [24] use capabilities and access lists, respectively. Capability-based access control is also explored in [11], while [18] and [30] describe solutions based on access lists. Access control in database systems is addressed in several papers, including [19, 7, 23]. Formal treatment of the problem can be found in [27, 3, 14, 29, 1]. Although some work has been performed on access control in the Internet and the World Wide Web (WWW) [5, 10], the specific problems mentioned above—especially partitions—have not received much attention. An important exception is [23], which presents an approach that handles communication and site failures by ensuring the eventual consistency of access control information that has been replicated across multiple machines. However, such guarantees may not be strong enough for all applications. In general, the problem of partitions has been addressed primarily in the context of databases [6, 20] and file systems [12]. While relevant, the specific characteristics of the access control problem often make these solutions inappropriate, as well as provide opportunities for optimization that are generally unavailable for ordinary database operations.

This paper addresses access control in wide-area networks where failures, recoveries, and temporary network partitions may be frequent and where services may be massively replicated.[1] In such systems, partitions and failures make it impossible to achieve all the desired properties for access control at all times, so any solution must make tradeoffs between security, availability, and performance. The main contributions of the paper are the identification of techniques for dealing with access control in the presence of partitions and an algorithm that allows application control over these tradeoffs.

2 System Model

2.1 Problem Statement

The access control problem can be characterized in terms of users attempting to access applications on host computers in a distributed system. We assume that each user is uniquely identified by a user id and that an authentication method is available to ensure that a message sent by a user U has indeed been sent by this user. Any public key cryptosystem, such as the RSA algorithm [22], could be used for this purpose. Further information on authentication can be found elsewhere [26, 8, 14, 28, 17].

A simple example of the access control problem would be a service that provides stock quotes, but only to those users who have paid for the service. A more complicated example would be a distributed information service that maintains data for an organization. In this case, some user identifiers could have been compromised or users terminated, so it is important to be able to prevent those users from accessing or changing information.

For each distributed application A, the set of hosts that run this service form the set *Hosts(A)*. The set of users that have right to use A, that is, have the right to send messages

*This work supported in part by the Office of Naval Research under grant N00014-96-0207.

[1] Although we focus here on wired networks, similar problems exist in mobile computing systems, so our solutions could be applied in this context as well.

0-8186-7813-5/97 $10.00

to the application, form the set *Users(A)*. Finally, the users that have the ability to change the access rights associated with *A* form the set *Managers(A)*. Where clear from context, we also use *Managers(A)* to refer to the set of hosts from which changes in access rights originate. For simplicity we only address two types of access rights: use and manage.

Given the emphasis on wide-area networks, the solution is constrained by the following assumptions. First, the number of different distributed applications may be very large. Second, each application may be replicated on a large number of hosts and may have a large number of users. However, we assume that the number of managers for a given application is relatively small and that the set of managers changes relatively infrequently. Finally, we assume that the frequency at which an application is used is much higher than the frequency at which a manager adds or revokes access rights.

In the area of failures, we assume that the protocol is required to deal with temporary network partitions and host failures and recoveries. Failures of individual hosts are assumed to be relatively rare (e.g., the MTTF of any individual host being on the order of several weeks [15]), but that temporary network partitions caused mostly by network congestion can be frequent. Finally, we assume that the hosts executing the manager portion of the protocol always provide correct information or do not provide any information at all, i.e., they only experience crash or performance failures.[2] Other hosts can experience any type of failure, but it is the responsibility of the application or user to detect these failures. The access control mechanisms can be then used to block access to applications from users on these hosts. These failures can include a malicious adversary gaining control of a host or a user identity.

Note that, although we are presenting the problem in terms of a user having the right to access an application, we could state it just as easily in terms of a host having the right to send a message to an application on another host. In this case, a host would be identified by its Internet address and a similar authentication scheme would be required.

2.2 Access control system components

The system components, illustrated in Figure 1, have the following functionality.

- **Network** provides (unreliable) point-to-point and multicast communication.
- **Access Control** checks whether each message received from the network can be forwarded to the intended application on this host, i.e., if the sender is authorized to use the application.
- **Access Control Management** stores the local copy of the current access control list and provides information to Access Control as needed.
- **Manager** is an application level entity that issues commands to change access rights.
- **Application** performs the user prescribed activity.

[2] The failure model of managers could be extended to Byzantine failures [13] by using ideas from secure membership protocols [21].

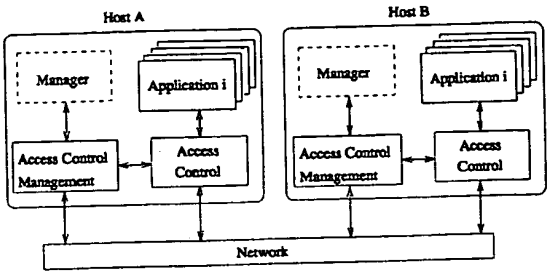

Figure 1: System components

Not all components need be present on all hosts; for example, as described above, managers are generally found on only a relatively small subset of hosts. Note also that in this design, the access control mechanisms encapsulate the application, essentially creating a wrapper that enables the application to be written without needing to address access control. Among other things, this allows access control mechanisms to be added transparently to existing applications.

The proposed protocol uses access control lists as the fundamental underlying mechanism. In this design, the access control management component maintains an access control list for each application that includes the users allowed to access the application, as well as the application's managers. Since each host has its local view of these sets, the problem of access control management becomes one of coordinating the views of different hosts.

2.3 Access control operations

For simplicity, assume that users access application *A* with operation **Invoke(A)**. Such an operation is allowed if the issuing user has the "use" right to *A*. Two operations are defined for managing access rights:

- **Add(A,U,R)** adds right *R* ("use" or "manage") to application *A* for user *U*.
- **Revoke(A,U,R)** removes right *R* to application *A* for user *U*.

In a centralized system, the semantics of these two operations are clear. The addition or revocation of access rights is expected to be synchronous (i.e., has taken effect when the invocation returns) and is usually performed quickly. Unfortunately, in a distributed system where communication delays are unpredictable and partitions and failures may occur, such goals are more difficult to achieve. However, it is possible to guarantee that the operations take effect eventually or, in some cases, within a known bounded time. For simplicity, we retain the same blocking invocation semantics, so that an operation is guaranteed to have taken effect throughout the system when the call returns. Of course, since these operations may block for a significant length of time, it would be useful in some cases to have non-blocking versions that return immediately.

Now, consider the possible properties of the access control operations. As mentioned, it is possible to implement

an operation so that it will eventually take effect at all hosts, or, in some cases, so that it takes effect within a known time bound. The second property is especially relevant for revoking rights, since it can provide some level of assurance against future unauthorized accesses.

While implementing the first option is possible since all partitions are eventually repaired, the second is more problematical. In particular, providing a time bound on revocation cannot be based on communication between the manager issuing the command and hosts executing the application because a network partition may last longer than any fixed time limit. Thus, timebound revocation must be based on passage of time, i.e., on having access rights expire after some amount of time unless refreshed by a manager. For access lists, this simply means removing an entry that is not refreshed within the specified amount of time. Of course, any time-based solution such as this can result in legitimate users being unable to use the application until the partition between the host and a manager has been fixed.

For the analogous issue of granting access rights, a similar timebound approach would allow a user to access an application after the host has been unable to contact a manager for a fixed period of time. Such an approach is unacceptably lenient for most applications, except perhaps certain Internet-based information or entertainment services where customer satisfaction is paramount and potentially unauthorized access results only in minor revenue loss. However, since network partitions can separate a legitimate user from all hosts running the application in any event, making a timebound guarantee is not possible in all situations.

In general, then, the tradeoff faced in the event of partitions is between availability and security, and no single policy is appropriate for all applications. For example, if the application provides confidential information, security is likely more important than availability and the system must be able to deny access to users whose identity has been compromised. Similarly, if the application allows users to purchase expensive merchandise or undertake significant financial transactions, it may be more important to be able to check that the user is still authorized to use the service than to grant access. At the other extreme, to ensure user satisfaction, availability can be more important than security for services such as on-line magazines and newspapers. Thus, any access control mechanism should be able to accommodate different policies and allow the application to choose different tradeoffs in the case of partitions.

3 Protocol

The purpose of the access control protocol is to provide hosts in *Hosts(A)* with information they can use to decide if messages arriving from the network should be forwarded to the application A. Realizing this functionality requires implementing two types of operations, those for changing access rights and those for checking rights when a message arrives. An inherent tradeoff exists between these two types of operations. If the operations that change rights distribute information to all hosts that execute a particular application, then checking only requires accessing local information. Of course, distributing this information to all the hosts can be costly, plus all hosts typically do not require information about all users. A second possibility is to disseminate the access control information just among the managers, in which case checking access rights at an application host requires communicating with at least one manager. A third option is to only change the information locally at the manager issuing the update operation, in which case checking access would in general involve communicating with all managers to locate the information.

The approach presented in this paper is similar to the second option, but with caching at application hosts to improve efficiency.

3.1 Basic protocol

In our solution, only the managers of a given application maintain complete access control information, with each host that executes the application caching part of this information. In particular, when a host checks a user's access rights with a manager, it caches this information to optimize subsequent accesses by the same user. In the following, we assume that each application A has its own set of managers, *Managers(A)*, and that this set is fixed and known to all the hosts in *Hosts(A)*. Access control of A is assumed to be independent of other applications. We also assume initially that a method exists for instantaneously updating the access control information at all the hosts in *Managers(A)*; this assumption is relaxed in Section 3.3.

In our protocol, each host in *Hosts(A)* maintains a cache of the access control list for A, which we denote as *ACL_cache(A)*. Specifically, *ACL_cache(A)* contains the access rights that have been granted for some subset of the users of A. This information is updated whenever the host receives authorization information from a manager, either in response to a query from the host or when a manager explicitly forwards revocation information to the host (see below).

The initial version of the algorithm used by a host is presented in Figure 2. Note that in this basic version, the host checks the access rights repeatedly with different managers until it receives a response. Note also that an attempt to remove a non-existent access right from *ACL_cache* is equivalent to a no-op.

The other part of the problem occurs when a manager updates the access control information for some user U. In this case, the initiating manager transmits a message to all other managers in *Managers(A)* with the appropriate operation. Upon receiving such a message, a manager updates its local version of the access control list. In addition, if the operation is a revocation, the manager forwards it to all hosts to which it has granted access permission for U. As shown in Figure 2, this causes the host to flush its cache, i.e., remove the access permission for U from *ACL_cache*. Each manager maintains a list of these hosts in a table for this purpose.

3.2 Partitions

To deal with partitions, consider first the case where all managers remain connected,[3] but some application hosts cannot communicate with managers due to partitions. In this case, revocation is a problem since a host in *Hosts(A)* that is caching access control information may be unreachable for an indefinite period of time. As discussed above, the only approach that guarantees that revocation occurs within a known time bound is to expire rights automatically, so cached information is dealt with in this way. That

[3] Note that this assumption is trivially satisfied when only one manager is used.

```
initial { ACL_cache(A) = ∅; }

when Revoke(A,U) from network {
    if U ∈ ACL_cache(A) then
        ACL_cache(A) -= U;
}

when Invoke(A,U) from network {
    if U ∈ ACL_cache(A) then
        allow access;
    else {
        pending = TRUE;
        while pending do {
            send query to a manager in Managers(A);
            if response before timeout then
                pending = FALSE;
        }
        if response = Add(A,U) then {
            ACL_cache(A) += U;
            allow access;
        }
        else
            reject access;
    }
}
```

Figure 2: Access control at application host.

is, if a revocation associated with user U is initiated at time t and the time bound on revocation is T_e, then the protocol guarantees that U cannot access the application after $t + T_e$. Moreover, this holds even if the managers are unable to reach all hosts that are caching this information at time t.

The implementation of such a time limit depends on the accuracy of the local clocks. Clock synchronization is naturally impossible given partitions, but fortunately clocks typically have a relatively constant rate. Consider the problem of measuring T_e time units on local clocks. Let $C_i(t)$ be the amount of real (perfect) time that has passed when local clock C_i has measured t time units. The assumption about approximately constant rate states that there exists a known constant b such that for any local clock C_i and time period t, $b * C_i(t) \geq t$ holds. That is, every local clock is at most b times slower than real time. In practice, b should be fairly close to 1. Given such a b, then, managers provide hosts with access control information that has an *expiration period* of $t_e = T_e/b$, or, in other words, a time interval measured on the local clock time after which the hosts discard the cached information. Thus, if an access right is revoked, it is guaranteed that no host will grant access after T_e time units have passed.

This scheme can be implemented by adding a timestamp to each access control tuple in *ACL_cache(A)*. When a message from a manager arrives with access control information and an expiration period of t_e, the timestamp $t + t_e$ is included with the cached information, where t is the current time on the local clock. Now, every time the access right is checked, the stored timestamp is compared against the local time and if it has expired, the access control tuple is removed and the access is rechecked with a manager. A periodic check of *ACL_cache* can also be used to eliminate entries of users who have not accessed the application recently, which can save memory and processing overhead.

An extended access control protocol for application hosts is presented in Figure 3. In this code segment, function *Time()* returns the time from the local clock. Function *lookup(ACL_cache(A),U)* returns user U's access control information for application A as a tuple *(U,limit)*, where *limit* is the expiration timestamp. Note that the algorithm must address the transmission delays to ensure the timeliness of revocations. First, it must address unpredictable message transmission delays since if delivery of an access control message $Add(A,U,t_e)$ from a manager is delayed significantly, a revocation may have already been issued before the access control message reaches the application host. This problem is eliminated in the figure by the host only accepting access control messages if they arrive before a timeout of a timer set at the time the query to a manager was sent. Second, the transmission time Δ must be factored into the expiration time. The exact calculation of Δ is omitted for simplicity but it is at most the time period from when the query was sent to when the corresponding response was received.

```
initial { ACL_cache(A) = ∅; }

when Revoke(A,U) from network {
    if U ∈ ACL_cache(A) then
        ACL_cache(A) -= U;
}

when Invoke(A,U) from network {
    Rec = lookup(ACL_cache(A),U);
    if Rec ≠ NULL then
        if Time() < Rec.limit then {
            allow access;
            return();
        }
        else
            ACL_cache(A) -= U;

    pending = TRUE;
    while pending do {
        send query to a manager in Managers(A);
        if response before timeout then
            pending = FALSE;
    }
    if response = Add(A,U,t_e) then {
        ACL_cache(A) += (U,Time()+t_e-Δ);
        allow access;
    }
    else
        reject access;
}
```

Figure 3: Extended access control protocol at application host.

An addition to the algorithm can be used to handle applications where availability is more important than security when partitions occur. Figure 4 illustrates such an addition, where the application-specific constant R denotes the maximum number of attempts to verify an access right before allowing access as default.

```
when attempt to verify access right has failed R times {
    allow access;
}
```

Figure 4: Rule for high-availability applications.

Up to now, we have assumed that the set of managers is fixed and known by all hosts running the application. This assumption can easily be eliminated by using a trusted name service that provides each host with the set of managers when requested. If the set of managers changes, a scheme similar to the time-based expiration of cached information can be used to trigger a new query to the name service.

3.3 Handling manager updates

Consider now the problem of implementing manager updates when managers may be partitioned from one another, and hence, unable to communicate. Specifically, assume that manager *m* is partitioned from the other managers. Given the current algorithm, if *m* decides to revoke an access right, the other managers will continue to grant access even after the expiration period of the revocation if they fail to receive the update in time. This situation is, of course, undesirable since it would violate the guaranteed time bound associated with revocation.

One possible solution is to apply a time based strategy similar to that used above for the *ACL_cache*. In this case, an *inaccessibility period* T_i is specified for each application, and all rights to the application are considered valid as long as all managers are accessible. However, should any manager remain inaccessible for longer than the specified period, all access rights are frozen and no responses are sent to application hosts until all managers are accessible again. Naturally, T_i and t_e must be chosen so that their sum is at most T_e, and care must be taken to account for clock rate differences at managers.

While this strategy ensures that the guarantee of time bounded revocation is retained, it has several significant disadvantages. For example, managers may be forced to expire all access rights and therefore make the application completely inaccessible if a manager remains inaccessible for a long period of time. This can occur even in the case where the manager has failed and cannot issue a revocation, since in general partitions are indistinguishable from failures.

Another approach that provides a balance between availability and security can be implemented using *quorums*. Read and write quorums are often used in databases, as well as general data replication strategies [25, 2, 9]. In this case, a *check quorum* C is given for each application, which specifies that the access control module on the application host must obtain access right information from C (out of a total of M) managers before allowing access to an application. The corresponding *update quorum* is then $M - C + 1$ managers, which ensures that every update for which a quorum has been obtained has been received by at least one manager in any check quorum. A manager issuing an update uses a persistent strategy in disseminating the operation, in the sense that it repeatedly transmits the update to every manager until it succeeds. Thus, the time when an update quorum is obtained is the first point at which a guarantee can be made about an operation, specifically, that at most T_e time units can pass before the operation will take effect relative to a user attempting to access the application. With this approach, the inaccessibility of a small number of managers does not prevent new access control operations from being issued nor access to the application in most cases.

The quorum-based strategy cannot provide perfect security and availability, but it makes the tradeoff between these two attributes explicit and application specific. On one hand, to guarantee that every operation takes effect on all hosts running the application, an update quorum must always be reachable. In most cases, such a guarantee is only possible if the update quorum is one, i.e., each manager can make the update independently. This, however, means that the check quorum C must be equal to M, so the failure of even one manager would make the application unavailable. On the other hand, to improve the availability of the application when managers are unreachable due to failures or partitions, the check quorum should be small, ideally one. This, however, would mean that it would be more difficult to get an update quorum, thereby potentially delaying the effect of access right changes. Most practical applications would choose C between these two extremes and thus, sacrifice either some of the security or availability in favor of the other. This tradeoff is analyzed in Section 4.1.

Given that the update quorum is likely to be greater than one for availability reasons, it may be impossible for a manager to obtain a timely update quorum in some situations. There are several factors that alleviate this problem, however. First, although an update is only guaranteed to take effect after the update quorum is reached, it may be sooner in practice since any managers that have failed cannot further propagate the old access rights to application hosts. Furthermore, temporary partitions caused by congestion are typically short-lived, which means that the update quorum will often be reached after only a small delay. Finally, if it takes too long to reach a quorum, external methods are always possible. For example, human operators could be notified to evaluate the situation and if appropriate, request that the update be entered manually at unreachable managers.

3.4 Host failures and recoveries

If a host in *Hosts(A)* fails, potential users of the application *A* simply have to locate a new host. A manager attempting to send a revoke message to such a host is unable to tell the difference between a partition and a site failure, but it can stop resending the message when the access right would have expired based on the time mechanism. Recovery as far as the access control information is concerned is very easy since *ACL_cache(A)* can simply be initialized to null and refilled using the normal algorithm for checking access rights.

The failure of a manager is equally easy to handle since hosts needing access control information can simply contact another manager. The handling of a revoke operation reduces to partition handling. In particular, a failed manager *m* will essentially create a logical partition since no other manager is aware of application hosts that cached access control information based on interactions with *m*.

However, the expiration mechanism will ensure that the access rights will still be revoked within a bounded time. For recovery, a manager only needs to ensure that it retrieves current access control information from other managers before responding to access right queries.

4 Discussion

4.1 Analysis

The protocol presented in this paper allows access control to be enforced in wide-area networks that experience frequent partitions, and host failures and recoveries. Instead of providing one policy that either sacrifices availability at the expense of security or security at the expense of availability when both are impossible, the protocol allows each application to make its own choices. The availability and security enforced by the protocol, as well as its performance, can be customized by adjusting the number of managers M, the check quorum C, the expiration time T_e, and the attempt count R.

The performance overhead of the access control algorithm is naturally $O(C/T_e)$, since the access rights have to be checked every T_e time units and checking them involves communication with at least C managers. Thus, increasing T_e reduces the overall overhead of the protocol, but also increases the potential delay when an access right is revoked.

The delay that the access control protocol imposes on an individual message addressed to an application is very small if the valid access control entry is already in the cache. If the entry is not in the cache, the delay is $O(C)$ in the normal case where at least C managers are accessible, but $O(R)$ if the required number are not accessible. Reducing R will naturally reduce this worst case delay, but at the cost of reduced security.

The security and availability for a given value of C depend heavily on the characteristics of the particular distributed system. Here, we demonstrate the calculation using a simplified model of the system where the probability of a site s_1 being inaccessible from site s_2, either because s_1 has failed or the network connecting these two sites has partitioned, is identical and independent for any two sites. Let this probability be denoted by P_i. For the analysis below, we assume $R = \infty$, thereby ensuring that access is only allowed if the check quorum of managers is reached.

Let P_A be the probability that a host is able to verify the access control information of a legitimate user in a timely fashion, i.e., P_A characterizes the availability of the application. Let P_S be the probability that a manager is able to revoke globally the access rights of a user in a timely fashion, i.e., P_S characterizes the security of the system. Let $P_A(C)$ and $P_S(C)$ be the values of these probabilities given a check quorum of size C in a system with fixed number M of managers.

Now, $P_A(C)$ can be calculated using the binomial distribution since $P_A(C)$ is simply the probability that at least C out of M managers are accessible to the host that issues the access control query. Thus, $P_A(C)$ is:

$$P_A(C) = \sum_{k=C}^{M} \binom{M}{k} (1 - P_i)^k (P_i)^{M-k}$$

$P_S(C)$ can be calculated in a similar fashion since $P_S(C)$ is the probability that the manager that issues a revoke operation can access at least $M - C$ managers out of the other $M - 1$ managers. Thus, $P_S(C)$ is:

$$P_S(C) = \sum_{k=M-C}^{M-1} \binom{M-1}{k} (1 - P_i)^k (P_i)^{M-1-k}$$

Figure 5 illustrates the general behavior of these probabilities as a function of C. Note that although security can be very low with C close to 1 and availability can be very low with C close to M, there is a relatively large range of values of C around $M/2$ where both availability and security are very close to 1.

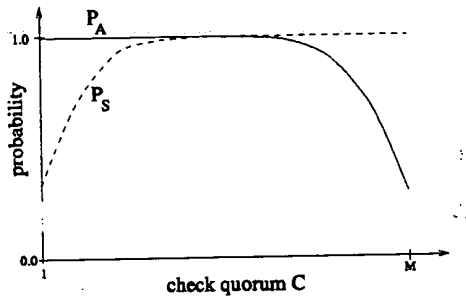

Figure 5: Availability and security curves

Table 1 lists values of these formulas given different values of P_i and C. In the table, we assume the number of managers is fixed at 10 and the check quorum C varies from 1 to 10. The table is calculated for P_i of 0.1 and 0.2. The table demonstrates that even with a high probability of site inaccessibility, both the availability and the security of applications can be high for values of C around $M/2$.

	$P_i = 0.1$		$P_i = 0.2$	
C	$P_A(C)$	$P_S(C)$	$P_A(C)$	$P_S(C)$
1	1.00000	0.38742	1.00000	0.13422
2	1.00000	0.77484	1.00000	0.43621
3	1.00000	0.94703	0.99992	0.73820
4	0.99999	0.99167	0.99914	0.91436
5	0.99985	0.99911	0.99363	0.98042
6	0.99837	0.99994	0.96721	0.99693
7	0.98720	1.00000	0.87913	0.99969
8	0.92981	1.00000	0.67780	0.99998
9	0.73610	1.00000	0.37581	1.00000
10	0.34868	1.00000	0.10737	1.00000

Table 1: Effects of C on availability and security.

Table 2 demonstrates the effect of the number of managers on availability and security. In this table, we vary the values of M and C given P_i values of 0.1 and 0.2. The upper part of the table illustrates that increasing M without increasing C is generally not a good idea since, although it increases availability, it decreases security. However, as

is illustrated in the lower part of the table, when C is increased at the same rate as M, both availability and security improve. Thus, if it is impossible to satisfy both availability and security goals given a set of managers, one way to solve the problem is to increase the cardinality of this set.

		$P_i = 0.1$		$P_i = 0.2$	
M	C	$P_A(C)$	$P_S(C)$	$P_A(C)$	$P_S(C)$
4	2	0.99630	0.97200	0.97280	0.89600
6	2	0.99994	0.91854	0.99840	0.73728
8	2	1.00000	0.85031	0.99992	0.57672
10	2	1.00000	0.77484	1.00000	0.43621
12	2	1.00000	0.69736	1.00000	0.32212
4	2	0.99630	0.97200	0.97280	0.89600
6	3	0.99873	0.99144	0.98304	0.94208
8	4	0.99957	0.99727	0.98959	0.96666
10	5	0.99985	0.99911	0.99363	0.98042
12	6	0.99995	0.99970	0.99610	0.98835

Table 2: Effects of M and C on availability and security.

In most realistic systems, site inaccessibility probabilities are much more heterogeneous than assumed above and furthermore, the probabilities are often dependent on one another since, for example, the failure of one communication link may make several managers inaccessible. Thus, detailed analysis is much more complicated. If the pairwise inaccessibility probabilities as well as the dependencies between these probabilities can be estimated, it is possible to calculate for each host the probability of reaching the check quorum and for each manager the probability of reaching the update quorum. The system availability and security can be estimated by averaging these probabilities. Furthermore, if the frequency of accesses at the hosts and the frequency of issuing access control operations at the managers are known, the average can be weighted using these frequencies to get a more accurate estimate of the overall system availability and security. Note that even if there is one manager that is frequently inaccessible from the others, the overall security of the system can be seriously reduced if this manager frequently issues and revokes access rights. Therefore, the assignment of managers to sites should be such that the inaccessibility between these sites is minimized.

4.2 Related work

As was pointed out in the introduction, the literature in access control typically does not address failures and partitions. One other approach to authorization that deals with site and communication failures in wide-area networks is described in [23]. Here, such events are dealt with by allowing changes in access control information to be updated eventually when communication has been resumed, with emphasis on eventual consistency. In contrast with our work, no guarantees are made on when the information will be updated nor do the algorithms make it possible for different applications to make different security versus availability tradeoffs.

The issue of providing time-limited access to information is also addressed by the idea of *temporal authorizations* [4]. With this technique, a user is granted access to an application (or object in general) for a known fixed period of time, typically on the order of days, weeks, or months. While similar to our time-based expiration approach, the goal of temporal authorizations is to allow a more detailed specification of users' access rights in the temporal domain, rather than addressing the security versus availability tradeoff in the presence of site and communication failures. Thus, these two aspects are orthogonal. It would be possible, however, to provide a course-grained simulation of our approach and guarantees by repeatedly providing short-lived temporal authorizations rather than granting permanent access rights.

5 Conclusions

This paper presents an approach to maintaining correct and consistent access control information in wide-area networks that frequently experience temporary partitions, and failures and recoveries of individual sites. Although similar in principle to the general problem of maintaining consistent information in distributed systems, the context imposes specific requirements and allows specific optimizations. We present an algorithm that uses time-based expiration and quorums to deal with partitions and site failures. Since maximal security and maximal availability can not be guaranteed at the same time in all cases, our algorithm allows each application to set the parameters that determine the level of security and availability, as well as the access control overhead. This approach is, we feel, preferable to the alternative of enforcing one chosen set of guarantees uniformly for all applications.

Acknowledgments

The initial form of the problem addressed in this paper was presented to us by Hilarie Orman and Sean O'Malley. Comments from the anonymous referees greatly improved the paper.

References

[1] M. Abadi, M. Burrows, B. Lampson, and G. Plotkin. A calculus for access control in distributed systems. *ACM Transactions on Programming Languages and Systems*, 15(4):706–734, Sep 1993.

[2] D. Agrawal and A. ElAbbadi. The generalized tree quorum protocol: An efficient approach for managing replicated data. *ACM Transactions on Database Systems*, 17(4):689–717, Dec 1992.

[3] G. Benson, I. Akyildiz, and W. Appelbe. A formal protection model of security in centralized, parallel, and distributed systems. *ACM Transactions on Computer Systems*, 8(3):183–213, Aug 1990.

[4] E. Bertino, C. Bettini, E. Ferrari, and P. Samarati. A temporal access control mechanism for database systems. *IEEE Transactions on Knowledge and Data Engineering*, 8(1'):67–80, Feb 1996.

[5] E. Bina, R. McCool, V. Jone, and M. Winslett. Secure access to data over the Internet. In *Proceedings of the 3rd International Conference on Parallel and Distributed Information Systems*, pages 99–102, Austin, TX, 1994.

[6] S. Davidson, H. Garcia-Molina, and D. Skeen. Consistency in partitioned networks. *ACM Computing Surveys*, 17(3):341–370, Sep 1985.

[7] N. Gal-Oz, E. Gudes, and E. Fernandez. A model of methods authorization in object-oriented databases. In *Proceedings of the 19th Conference on Very Large Databases*, pages 52–61, Aug 1993.

[8] L. Gong. Securely replicating authentication services. In *Proceedings of the 9th International Conference on Distributed Computing Systems*, pages 85–91, Jun 1989.

[9] M. Herlihy. Dynamic quorum adjustment for partitioned data. *ACM Transactions on Database Systems*, 12(2):170–194, Jun 1987.

[10] J. Kahan. A capability-based authorization model for the World-Wide Web. *Computer Networks and ISDN Systems*, 27(6):1055–1064, Apr 1995.

[11] R. Kain and C. Landwehr. On access checking in capability-based systems. *IEEE Transactions on Software Engineering*, SE-13(2):202–207, Feb 1987.

[12] P. Kumar and M. Satyanarayanan. Log-based directory resolution in the Coda file system. In *Proceedings of the 2nd International Conference on Parallel and Distributed Information Systems*, pages 202–213, San Diego, CA, 1993.

[13] L. Lamport, R. Shostak, and P. M. The Byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, Jul 1982.

[14] B. Lampson, M. Abadi, M. Burrows, and E. Wobber. Authentication in distributed systems: Theory and practice. In *Proceedings of 13th ACM Symposium on Operating Systems Principles*, pages 165–182, Oct 1991.

[15] D. Long, A. Muir, and R. Golding. A longitudinal survey of Internet host reliability. In *Proceedings of the 14th Symposium on Reliable Distributed Systems*, pages 2–9, Bad Neuenahr, Germany, Sep 1995.

[16] S. Mullender and A. Tanenbaum. The design of a capability-based distributed operating system. *The Computer Journal*, 29(4):289–299, Mar 1986.

[17] B. Neuman and T. Ts'o. Kerberos: An authentication service for computer networks. *IEEE Communications Magazine*, 32(9):33–38, Sep 1994.

[18] R. Oliver. Protection in a distributed document processing system. *ACM Operating Systems Review*, 24(2):56–65, Apr 1990.

[19] F. Rabitti, E. Bertino, W. Kim, and D. Woelk. A model of authorization for next-generation database systems. *ACM Transactions on Database Systems*, 16(1):88–131, Mar 1991.

[20] K. Ramarao. Commitment in a partitioned distributed database. In *Proceedings of the International Conference on Management of Data*, pages 371–378, Jun 1988.

[21] M. Reiter. A secure group membership protocol. *IEEE Transactions on Software Engineering*, 22(1):31–42, Jan 1996.

[22] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21:120–126, Feb 1978.

[23] P. Samarati, P. Ammann, and S. Jajobia. Maintaining replicated authorization in distributed database systems. *Data & Knowledge Engineering*, 18(1):55–84, Feb 1996.

[24] M. Satyanarayanan. Integrating security in a large distributed system. *ACM Transactions on Computer Systems*, 7(3):247–280, Aug 1989.

[25] D. Skeen. A quorum based commit protocol. In *Proceedings of 6th Berkeley Workshop on Distributed Data Management and Computer Networks*, pages 69–80, Berkeley, CA, Feb 1982.

[26] J. Steiner, C. Neuman, and J. Schiller. Kerberos: An authentication service for open network systems. In *USENIX Conference Proceedings*, pages 191–202, Dallas, TX, Winter 1988.

[27] S. Stepney and S. Lord. Formal specification of an access control system. *Software – Practice and Experience*, 17(9):575–593, Sep 1987.

[28] T. Woo and S. Lam. Authentication for distributed systems. *IEEE Computer*, 25(1):39–52, Jan 1992.

[29] T. Woo and S. Lam. Authorization in distributed systems: A new approach. *Journal of Computer Security*, 2:107–136, 1993.

[30] T. Woo and S. Lam. A framework for distributed authorization. In *Proceedings of the 1st ACM Conference on Computer and Communication Security*, pages 112–118, Fairfax, VI, Nov 1993.

Efficient large-scale access control for Internet/intranet information systems

Qi Lu Shang-Hua Teng
IBM Almaden Res. Center, San Jose, CA;

This paper appears in: **System Sciences, 1999. HICSS-32. Proceedings c 32nd Annual Hawaii International Conference on**
Meeting Date: 01/05/1999 -01/08/1999
Publication Date: 1999
Location: Maui, HI , USA
On page(s): 9 pp.-
Volume: Track5, References Cited: 13
Number of Pages: liii+341
INSPEC Accession Number: 6182239

Abstract:
Access control is a key problem for information processing, especially in a distributed environment such as the Internet and intranet, where a large amc diverse information resources within an enterprise will be made available for groups of diverse users to query. Information documents such as technology secrets and personal records are sensitive and should be accessible to a selec group of users based on their position in a company or an organization or eve based on how much the user is paying to maintain his/her right for informatic access. The access control problem, informally, is to determine which user is allowed to access what information. Access control for Internet information processing, in contrast to access control in a traditional operating system, ha: higher demand in dealing with a much larger scale problem in real time, due large amount of information and number of users in the Internet/intranet environment. We present an efficient method for the access control problem i which there are a large number of users and access groups. The main ingredi our method is a representation of a hierarchical access group structure in ter intervals over a set of integers and a decomposition scheme that reduces any group structure to ones that have interval representations. The interval representation allows the problem for checking access rights to be reduced tc interval containment problem. We use the interval tree, a popular data struct computational geometry, to efficiently execute the access-right checking met

Index Terms:
Internet authorisation information resources intranets tree data structures Internet access-right checking method data structure decomposition scheme enterprise hierarchical access group structure information access information processing inform resources information systems interval containment problem interval tree intranet !

scale access control operating system personal records real time

Documents that cite this document
Select link to view other documents in the database that cite this one.

SEARCH RESULTS [PDF Full-Text (224 KB)] PREVIOUS NEXT DOWNLOAD CITATION

Home | Log-out | Journals | Conference Proceedings | Standards | Search by Author | Basic Search | Advanced
Join IEEE | Web Account | New this week | OPAC Linking Information | Your Feedback | Technical Support | Ema
No Robots Please | Release Notes | IEEE Online Publications | Help | FAQ| Terms | Back to Top

Secure Web-based monitoring and control system

Furuya, M. Kato, H. Sekozawa, T.
Syst. Dev. Lab., Hitachi Ltd., Kawasaki;

This paper appears in: **Industrial Electronics Society, 2000. IECON 2000 26th Annual Confjerence of the IEEE**
Meeting Date: 10/22/2000 -10/28/2000
Publication Date: 2000
Location: Nagoya , Japan
On page(s): 2443-2448 vol.4
Volume: 4, References Cited: 3
Number of Pages: 4 vol.(xxviii+2997)
INSPEC Accession Number: 7274733

Abstract:
Recently, we have been seeing more and more examples of Web-based moni and control systems (WMCSs) with the widespread use of Internet technologi the Internet, however, systems or network services are subject to damage fr various threats. In this paper, we discuss a secure WMCS whose purpose and objective is to allow operations by authorized users via an intranet, while exc unauthorized operations. A very important component to protect a control ne or devices in a WMCS is a firewall that is called a "plant firewall" (PFW). We propose the main functions of a PFW for securing a WMCS. Those functions a user authentication, operation rights management, command filtering, progre reporting, communication path encryption, access logging and format conver

Index Terms:
Internet authorisation computerised control computerised monitoring information resources intranets system monitoring telecommunication security Internet World Web-based control system World Wide Web-based monitoring system access log authorized users command filtering communication path encryption format conversio intranet network services operation rights management plant firewall progress repo secure system systems damage threats unauthorized operation user authentication

Secure Web-based Monitoring and Control System

M. Furuya, H. Kato and T. Sekozawa

Systems Development Laboratory, Hitachi, Ltd.
1099 Ozenji, Asao-ku, Kawasaki 215-0013 Japan
Email:furuya@sdl.hitachi.co.jp

Abstract

Recently, we are often seeing more and more examples of Web-based Monitoring and Control System (WMCS) with the widespread use of the Internet technologies. In the Internet, however, systems or network services are subject to damage from various threats. In this paper, we discuss about a secure WMCS of which purpose and objective is to allow operation by authorized users via an intranet, while excluding unauthorized operation. A very important component to protect control network or devices in WMCS is a firewall that is called Plant FireWall (PFW). We propose main functions of PFW for securing the WMCS. Those functions are user authentication, operation right management, command filtering, progress report, communication path encryption, access log and format conversion.

1 Introduction

Key systems such as plant monitoring and control systems and production line systems have conventionally been constructed as closed systems, using the special infrastructures and communication protocols of the industry or vendor. With the recent widespread use of the Internet technologies, however, we are seeing more and more examples of systems that are being constructed with connections to control network and information network based on Internet technologies [1].

Such connections bring about many advantages to the user, such as increased bandwidth, increased distance, data sharing, and data provision for monitoring and control systems. Furthermore, there are plans for cooperation among multiple key systems, and cooperation with facilities management systems, maintenance and inspection systems, and other such information systems, and future development will probably continue in the direction of on-line maintenance, outsourcing services, Electronic Commerce (EC), etc.

However, connection to an open network and the use of universal technology presents new problems that did not exist with the conventional design and construction of key systems. The most serious of those new problems concern data security. As everyone knows, the world of the Internet is a place where services are increasingly subject to damage from various threats, such as unauthorized access, wiretapping or tampering with private data, system failures caused by viruses, denial of service due to network and server overload. What is making this problem worse is there is no such thing as security countermeasures that are 100% perfect and new techniques for attack are constantly being devised. Furthermore, differently from physical attacks, for attacks on information systems there are no temporal or spatial constraints and if computers and their peripheral devices are present, attack is possible, there is little awareness of criminal activity, and the means of identifying the attacker are few.

We discuss about a monitoring and control systems that are connected to a control network and information network. We call the system Web-based Monitoring and Control System or WMCS. In WMCS, a very important component to protect control network or devices is Plant FireWall (PFW).

Here, We first describe a security policy, a security concept and secure system architecture in WMCS.

0-7803-6456-2/00/$10.00 ©2000 IEEE

We then explain about main functions on the PFW. Finally, We give you general notes when building a secure WMCS.

2 Data Security Measures

2.1 Security Policy

Internet technology has created a need for security as well as for convenience in use. The WMCS, being no exception, also requires security measures. In particular, unauthorized access to the monitoring and control systems that are key systems in industrial plants and production lines, such as opening or closing the gates of river dams and shutting down power generation facilities for example, is a very serious matter.

Usually, the boundary between an intranet and the Internet is protected by a firewall, so the tendency is to consider such intranets to be safe. Because of that, we see examples of WMCS being implemented without any particular consideration given to the establishment of data security measures. The reality is, however, that intranets are not entirely safe from illicit access and, what is more, network crimes perpetrated from within by employees has been on the increase. Countermeasures are indeed needed. The purpose and objective of aWMCS is to allow operation by authorized users via an information network, while excluding unauthorized operation.

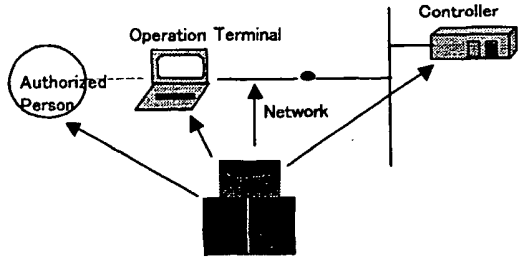

Fig.1 Attack Target

2.2 Attack Target

As shown in Fig. 1, the targets for malicious attacks on a WMCS can be classified into the following four categories.

(1) Directly attack controllers and devices that are on the control network.
(2) Monitor the information network to obtain information that would suggest methods of attack.
(3) Gain access to an authorized operation terminal and use it to make an indirect attack or convert a terminal into an authorized operation terminal to make an attack.
(4) Obtain information that suggests a method of attack from an authorized person by social engineering or become an authorized person and make an attack.

Of the above four classes of targets, the social engineering of (4) requires that sufficient attention be given to authorized personnel from an early time, while the others require the implementation of technically adequate countermeasures.

2.3 Target Protection

The main measures for protecting the targets described above are listed below.

(1) Establish sufficient access restrictions to controllers and devices that are on the control network. Or restrict access to the entire control network at a firewall.
(2) Use encrypted communication within the information network.
(3) Establish sufficient access restrictions to operation terminals. In particular, it is necessary to pay careful attention to whether or not monitoring and control programs can be used as platforms for attack. It is also necessary to confirm that no unauthorized program has been placed in the operation terminal. Of the unauthorized programs, those that attack other equipment by means of communication (that communication is taking place is not evident) require attention. With these attack programs, the perpetrator cannot be traced by analysis of the log on the control network side alone, so the likelihood of their being used by malicious persons is extremely high.
(4) Countermeasures for malicious terminals becoming the creation of authorized operation terminals are needed. When the only countermeasure is IP address filtering on the control network side alone, there is virtually no protection against attacks by falsification of IP addresses.
(5) Countermeasures for malicious persons becoming authorized users are required. Usually, user authentication by means of

user ID and password sets is performed. But they are susceptible to attack by social engineering or password cracking and the possibility of information leaks is also high, so a higher level of safety can be achieved through user authentication by means of a Public Key Infrastructure (PKI) that employs IC cards.

3 Secure WMCS

The three elements to achieving a secure WMCS are user authentication (including qualifications if possible), protection of what can be operated, and encrypted communication. At the very least, these three elements must be included in a secure implementation.

3.1 User Authentication

User authentication is usually accomplished by mean of a combination of user IDs and passwords, but there are cases in which this is no sufficient. Passwords are generally combinations of eight or so alphanumerical characters, but most protocols pass them on as plain text over the network, so they are easily broken by wire-tapping. Also, encryption of the password alone still permits simple cracking by means of a dictionary attack, so the password should be encrypted together with other messages. This measure alone is also weak against a replay attack (attack by sending the same cipher text), so a one-time password protocol is a good measure.

For user, operation terminal, command or event authentication in a WMCS, the public key encryption system has been proposed. Public key encryption involves a number of methods (levels).

(1) A password encrypted with the target's public key on the user side is authenticated on the target side.

(2) A challenge code given by the target side is digitally signed with the user's private key on the user side, and the signed code is authenticated on the target side.

(3) The procedure described in (2) above is performed on both the user side and the operated side.

(4) Procedure (2) or procedure (3) is authenticated on the basis of a certificate that is issued by a trusted Certification Authority (CA).

In particular, the PKI in which a CA is established and conducts certificate-based authentication as described in procedure (4) above is most often employed in areas such as the Secure Socket Layer (SSL) certificates that are used by Web browsers and the EC Secure Electronic Transactions (SET) certificates. In future, we must continue to consider PKI-based operation in WMCS as well.

3.2 Protection Approach

There are two approaches to the protection of operated equipment (controllers and device). One is to protect the control network with a single firewall, the other is to give each unit of operated equipment its own individual protection.

For key systems such as plants and production lines, many units of operated equipment are used and they are each designed for compactness in consideration of limited resources, so they are not easily incorporated into security programs. Furthermore, individual protection of controlled units of equipment does not protect the amount of traffic on the control network, which allows invasion of the control network. Protection by a firewall is suitable for large-scale plant network environments.

3.3 Encrypted Communication

Wire-tapping on an information network is easy to accomplish, so encrypted communication is a necessity. In particular, it is dangerous if the private data required for authentication, operation commands, data related to the operated equipment or other such data are wire-tapping. Accordingly, encrypted communication is employed on the information network.

When commands are transmitted, high responsiveness is required, so common key encryption is used as the encryption method. There are two methods for sharing the common key on the sending side and the receiving side. One is providing both sides with the key in advance and another is sharing the session key by using public key encryption. The latter is used for management

and convenience

4 System Architecture

Fig.2 shows a system architecture based on an security policy. We recommend protecting the control network with a single firewall and call the firewall Plant FireWall (PFW). A Secure WMCS is realized by using PFW as a connection component. [2]

The PFW that we propose is an application gateway. That is to say, data sent from an operation terminal is analyzed and only proper data is sent on to the control network. Naturally, because the analysis of data content creates a processing load, the application gateway is used together with a packet filter type firewall to protect against denial of service attacks.

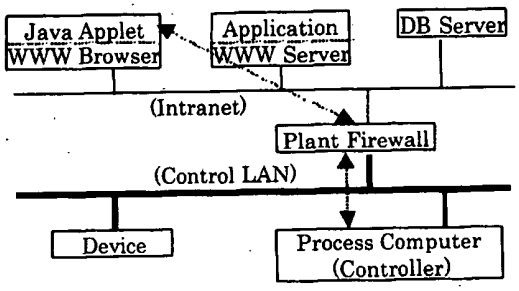

Fig.2 System Architecture

5 Security Functions

5.1 User Authentication

User authentication is currently done by means of user ID and password encrypted with PFW public key on the user side. PFW manages all passwords for authorized users. However, implementation of PKI-based user authentication is planned for the future.

5.2 Operation Right Management

In a WMCS, the management of operation right is an important function. Usually, there are multiple authorized users, and problems arise when those users perform operations at the same time, as they please. If operation right management is not implemented, operation will be governed by the last-in-first-out principle (the operation of the user that executed the operation even slightly later in time is effected).

Therefore, we introduce the concept of operation right. The one user that has the operation right can perform operations consecutively with priority. Even an authorized user cannot take on the operation right except at times when no other authorized user is performing an operation. There are also functions for requesting the operation right, transferring the right, automatic expiration of the right, compulsory deprivation of the right, etc. The operation right is granted in units of operated equipment.

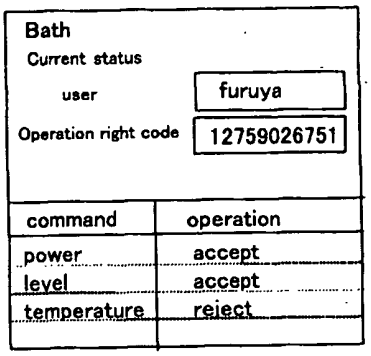

Fig.3 Command Filtering Table

5.3 Command Filtering

Within the PFW, there is a command filtering table for each unit of operated equipment. The command filtering table can specify whether an operation command is to be accepted or rejected according to the command, effective parameter range, user qualifications, and time period (Fig. 3). In command filtering, the default rule is to reject and operation commands for operated equipment for which there is no setting in the table are ignored.

The unit of operated equipment that is used in the command filtering table is a unit that is suitable for management of the operation right, whereas the setting of the filtering operation for each operation command reflects the security policy of the organization that constructs the WMCS. For example, for a bath fixture as operated equipment, settings such as "Commands for the water level adjustment system are accepted, but commands for the water temperature adjustment system are

rejected." may differ according to the policy of each organization.

5.4 Progress Report

Basically, the main operations in a WMCS involve a change in a setting value or a switch position. The implementation of real-time feedback control that includes the operation terminal is unrealistic at this time. Even in the changing of settings, the change is not necessarily effected immediately after the operated equipment (controller) receives the command. If the user must wait with no indication of the progress of the operation during the time between when the command is given and when it is effected, an error in operation may result. Therefore, the PFW sends a progress report to the operation terminal while connected to the operated equipment.

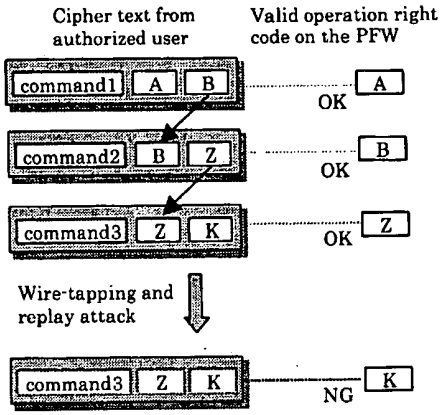

Fig.4 Variable Operation Right Code

5.5 Communication Path Encryption

At the time of user authentication, the session common key that is used when the commands or other such data is transmitted is shared dynamically, and communication path encryption is performed between the operation terminal and the PFW with the PFW public key. It is not sufficient, however, to conduct the communication by simply encrypting the transmitted commands, etc. Assume, for example, that the cipher text for the valid data is "ZY." Even if an attacker is not able to decipher "ZY," it is still possible counterfeit the plain text header of the packet and re-send "ZY," which would be valid. To prevent this, a one-time password scheme in which the cipher text of the valid data "AB" is changed each time, and once the cipher text is received, it is not accepted a second time. The one-time password scheme employs time stamps or variable operation rights.

The principle of the variable operation right protocol is illustrated in Fig. 4. In this protocol, when there is an operation on the operation terminal, a random operation right code is generated, and a pair of codes, that are valid for this time and the next time, is added to the command and parameters. When communication between the operation terminal and the PFW is successful, the operation right code is updated to the next valid code on both sides. Even if an attacker obtains the cipher text on the communication path and uses it in a replay attack, that code differs from the current valid the operation right code that is managed on the PFW side, so a simple replay attack cannot result in an unauthorized operation.

5.6 Access Log

At the PFW, all events are logged for the purpose of tracking and investigation to determine who performed an operation, whether an unauthorized operation has been executed, and so on. Because the application gateway method is employed, even data is logged and the processing load is high. The items that should be logged and the time period for which the log data is saved require separate study.

5.7 Format Conversion

In most cases, information networks and control networks use different formats and protocols, so the PFW must perform format conversion. Usually, a special format that can be understood both by the operation terminal and the PFW is used on the information network. However, [3] is trying XML (eXtensible Markup Language) format as a universal format. We are going to use XML format for the future, too.

6 Conclusion

We have made specific proposals that mainly concern a data security countermeasure and a plant firewall as an approach to a solution for security

problems in a Web-based Monitoring and Control System. Below we list a few final addenda.

(1) Excessive data security measures not only create inconvenience, but also increase cost. It is important to take technical and operational countermeasures that are based on a systematic policy.

(2) Because new threats and methods of attack are constantly appearing, data security operations must involve a continuous four-phase cycle of countermeasure → guarantee → detection → investigation.

(3) For control networks, too, there have recently been many attempts to utilize the same protocols that are used in information systems, with the objective of achieving universal and open systems, but attention must be given to the increased security risks that result. Specialized protocols that have private specifications are more difficult to attack directly.

(4) Care must be taken to prevent bypath. Even when tight security measures are in place, even the smallest bypath may permit unauthorized invasion. Remote access servers that are easily set up for the purpose of remote maintenance or other such purposes require particular attention.

(5) At this time, the implementation of a WMCS via the Internet remains insecure. First of all, a full evaluation should be made within the confines an intranet that is isolated from the Internet by a firewall.

References

[1] A.C.Weaver, "Monitoring and Control Using the Internet and Java", in Proc. IECON '99, pp.1152-1158, 1999

[2] M.furuya, "WWW-Browser-based Monitoring System for Industrial Plant", in Proc. IECON '99, pp.1146-1151, 1999

[3] M.Munson, "Flexible Internetworking of Devices and Controls", in Proc. IECON '99, pp.1139-1145, 1999

SEARCH RESULTS [PDF Full-Text (1088 KB)] PREVIOUS NEXT DOWNLOAD CITAT

Browser-style interfaces to a home automation network

Corcoran, P.M. Desbonnet, J.
Dept. of Electron. Eng., Univ. Coll. Galway;

This paper appears in: **Consumer Electronics, IEEE Transactions on**
Meeting Date: 06/11/1997 -06/13/1997
Publication Date: Nov 1997
Location: Rosemont, IL , USA
On page(s): 1063-1069
Volume: 43, Issue: 4
ISSN: 0098-3063
References Cited: 5
CODEN: ITCEDA
INSPEC Accession Number: 5804791

Abstract:
The design and implementation of a browser-style interface to a home autom network is described. The interface supports browsing and navigation of netw devices and context structures and the user can interact with individual devic the network and access and control content and object structures within thes devices. The interface can be used to access a local home automation networ from a standard desktop PC, or from a TV/set-top box system with built-in interface hardware. Furthermore, as the interface is implemented using conventional Internet and home network technology it can provide access an control services to the home network from any computer with an Internet connection

Index Terms:
Internet consumer electronics graphical user interfaces home automation telecontrc transport protocols CEBus Internet connection Internet technology TV system brov style interfaces built-in interface hardware consumer electronic bus context structure design home automation network implementation navigation network devices obje structures set-top box system

Documents that cite this document
Select link to view other documents in the database that cite this one.

SEARCH RESULTS [PDF Full-Text (1088 KB)] PREVIOUS NEXT DOWNLOAD CITATION